This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2019.2930521, IEEE Transactions on Cognitive Communications and Networking



# Deep Reinforcement Learning for Intelligent Internet of Vehicles: An Energy-Efficient Computational Offloading Scheme

Zhaolong Ning, *Senior Member, IEEE*, Peiran Dong, Xiaojie Wang, Lei Guo, Joel J. P. C. Rodrigues, *Senior Member, IEEE*, Xiangjie Kong, *Senior Member, IEEE*, Jun Huang, *Senior Member, IEEE*, Ricky Y. K. Kwok, *Fellow, IEEE*

***Abstract*—The emerging vehicular services call for updated communication and computing platforms. Fog computing, whose infrastructure is deployed in close proximity to terminals, extends the facilities of cloud computing. However, due to the limitation of vehicular fog nodes, it is challenging to satisfy the quality of experiences of users, calling for intelligent networks with updated computing abilities. This paper constructs a three-layer offloading framework in intelligent Internet of Vehicles (IoV) to minimize the overall energy consumption while satisfying the delay constraint of users. Due to its high computational complexity, the formulated problem is decomposed into two parts: flow redirection and offloading decision. After that, a deep reinforcement learning based scheme is put forward to solve the optimization problem. Performance evaluations based on real-world traces of taxis in Shanghai (China) demonstrate the effectiveness of our methods, where average energy consumption can be decreased by around 60 percent compared with the baseline algorithm.**

***Index Terms*—Internet of vehicles, deep reinforcement learning, computation offloading, energy efficiency.**

## I. Introduction

Human driving mistakes and misjudgments lead to 90% of traffic accidents [1]. Internet of Vehicles (IoV) is a key enabling technology for smart cities, and is promising to alleviate congestion and reduce traffic accidents caused by driving misbehaviors. With the development of IoV and vehicle intelligence, vehicles are transforming from travel tools to intelligent terminals [2]. For an American, over ten hours are spent in vehicles each week on average. Thanks to the telematics applications, IoV is becoming a platform for convenience and entertainment during the journey. However, the ever-increasing requirements of users' Quality of Experience (QoE) and vehicle intelligence challenge the implementation of telematics applications, especially the computation intensive applications, such as complicated decision making and real-time resource management [3].

Cloud computing, which evolves from distributed computing to grid computing is developed to overcome above obstacles of vehicular applications. Although it can provide sufficient computing resources, cloud or cloudlet centers may be far away from terminal users, reulting in large network latency [4]. In 2020, more than 1.5 billion vehicles will be connected, and the requirements of real-time computing processing and data management challenge the bandwidth of core networks and users' QoE [5]. Distinct sorts of data in IoV are collected and processed by vehicles to support the decision-making process of traffic management based applications, such as vehicular safety and intelligent driving [6].

Mobile subscriber is becoming the dominated data source, for example, a self-driving vehicle can generate more than 1 TB data per hour. How to make full use of the computing and storage abilities of vehicles is rather challenging, thus mobility supporting and geographical distribution are crucial. Motivated by this observation, the concept of fog computing is presented by Cisco company. By deploying fog servers with computational capabilities and storage resources at the edge of networks, users' QoE, especially network latency requirement, can be significantly promoted. According to the report of Cisco global cloud index, the total generated amount of data by Internet of Things (IoT) devices will be 4-fold greater in 2021 than that in 2016, and the amount of data stored on the edge of devices will reach over 80% of the total created data in 2021 [7]. The widely distributed and real-time connections call for fog computing, because distinct communication patterns, e.g., Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), and Vehicle-to-Sensor (V2S), coexist in IoV.

With the hypergrowth of sensors in intelligent vehicles, the variety and vast amount of generated data significantly challenge the decision making and network management in IoV, and call for intelligent computing technologies [8]. The integration of emerging technologies, such as Deep Reinforcement Learning (DRL), computational offloading and cognitive computing, is promising to widen the horizons of solutions in IoV [9]. Based on the report by McKinsey & Company in 2016, intelligence and connectivity are essential for autonomous vehicles, which will occupy 15% of the vehicle market in 2030 [10]. Generally speaking, the superiorities of intelligent IoV can be summarized as: cognitive intelligence,

Z. Ning, P. Dong, X. Wang and X. Kong (Co-corresponding author) are with the School of Software, Dalian University of Technology, 116620, Dalian, China. Email: xjkong@ieee.org.

Z. Ning, L. Guo (Co-corresponding author) and J. Huang (Co-corresponding author) are with the Chongqing Key Laboratory of Mobile Communications Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China. Email: guolei@cqupt.edu.cn and jhuang@cqupt.edu.cn.

Z. Ning and R. Kwok are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong.

Joel J. P. C. Rodrigues is with the National Institute of Telecommunications (Inatel), Santa Rita do Sapucaí, MG, Brazil; Instituto de Telecomunicações, Portugal; Federal University of Piauí (UFPI), Teresina, PI, Brazil.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2019.2930521, IEEE Transactions on Cognitive Communications and Networking



reliable decision making, efficient resource utilization, and rich market potentiality.

### A. Motivation

In 2016, the power consumption of data centers in China has reached 120 billion kilowatt-hours, dominating around 1.5% of the total electricity consumption in China and is even larger than the gross generated power by the Three Gorges Project. Althougn fog computing is acknowledged to be an alternative to reduce network latency and energy consumption in IoV, energy shortage is becoming a crucial obstacle to limit the development of IoV. According to the statistics [11], the increasing energy demands are 16.5% for electricity, 11.9% for traditional renewable, and 6.4% for coal. Instead, fossil fuels make up 49.4% of the total energy consumption, which are mainly consumed by current vehicles. Therefore, it is worth to carefully consider the energy consumption of the vehicular applications. Specially, Electric Vehicles (EVs) or hybrid EVs will dominate the market in the very near future. This motivates us to consider energy consumption in IoVs.

Current researches on green IoV mainly focus on energy management of either battery-equipped Road Side Units (RSUs) or EVs. However, energy-efficient computational offloading has not been fully investigated, which is significant to save energy and accelerate the computation process in IoV. Because of the limited computing and hardware capabilities of vehicles, two issues should be fully considered for fog-enabled offloading in IoV: one is whether offloading tasks by fog computing or not; the other is whether offloading network loads fully or partially. Therefore, energy-efficient computing based on the coordination between cloudlet servers and fog nodes deserves to be well studied.

With the objective of realizing intelligent energy-efficient IoV, how to implement DRL algorithms in IoV systems deserves to be well investigated. Data centers with strong processing abilities and large memory resources are suitable for employing DRL algorithms. In order to provide (approximate) real-time responses for vehicles, implementing DRL algorithms at the network edge is regarded as a promising solution, since it can process the generated data close to the event occurrence location. However, the majority of the machine learning and artificial intelligence algorithms consumes a large amount of energy, which may not always be available for edge devices. Therefore, RSUs equipped with MEC servers are desired to undertake the decision making task.

### B. Contribution

Different from the emergent message transmission (such as road safety and dangerous activities) in IoVs, our work focuses on the services for vehicular entertainment, which are not urgent but consume large resources, such as bandwidth and energy. In order to minimize the overall energy consumption of the computational facilities and vehicles while satisfying the delay constraint for traffic offloading, this paper constructs a three-layer energy efficient offloading scheme in IoV. The designed framework is rational for intelligent IoV, because computational intensive tasks can be processed on fog nodes to satisfy the requirements of delay-sensitive vehicular applications. After that, a DRL based scheme is designed to solve the optimization problem. The main contribution of our work can be summarized as follows:

- We construct a three-layer architecture in IoV, and formulate an energy-efficient computational offloading problem to minimize the overall power consumption while satisfying the delay constraint.
- The formulated problem is decomposed into two parts, i.e., flow redirection and offloading decision. For the first part, an Edmonds-Karp based scheme is developed to balance the traffic load among RSUs, while a DRL based model is investigated for the second part.
- The DRL model is constructed based on the queuing theory, where the system state comprises the arrival rate of moving vehicles, the arrival rate of offloading task folws and the number of parked vehicles. These variables are the main factors that affect the offloading decision macroscopically.
- Performance evaluations based on the real-world traces of taxies in Shanghai (China) demonstrate the effectiveness of our solutions, i.e., the developed Edmonds-Karp algorithm can balance the load of RSUs with relatively low energy consumption, and the DRL based offloading decision scheme is superior to other compared methods with various parameters.

The rest of this paper is organized as follows. We first provide an overview of the related studies in Section II. After that, we illustrate the system model in Section III. The optimization problem is formulated in Section IV. We briefly introduce the overview of DRL in Section V. Edmonds-Karp and the DRL based schemes are specified for energy-efficient offloading in IoV in Section VI. In Section VII, performance evaluations demonstrate the effectiveness of our methods. Finally, we conclude our work.

## II. Related Work

This section outlines the current research progresses of intelligent computing for IoV from the following three aspects.

### A. Cognitive IoV

By data mining, pattern recognition, and natural language processing, cognitive computing is an alternative for decision making. As an integration of cognitive computing and IoV, cognitive IoV leverages cognitive technologies and comprehensively analyzes the collected data in physical and network spaces [12]. Different from the traditional IoV focusing on traffic data, cognitive IoV concentrates on the top-level issues, such as promoting the intelligence of vehicles for decision making. Furthermore, the spectrum scarcity of IoV is severe due to the ever-increasing density of vehicles [13]. An opportunistic traffic offloading scheme is investigated in IoV [14], which comprehensively takes users' satisfaction, offloading performance and network operators' revenue into consideration. A cognitive method is designed to predict offloading potential and access costs. Although current machine learning solutions generally concentrate on the computational efficiency,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2019.2930521, IEEE Transactions on Cognitive Communications and Networking



security is critical for the machine learning based computing architectures to ensure that the system can perform as expected without harmful behaviors. However, the high-level security mechanism generally accompanies large energy consumption due to the high computational complexity, challenging the energy-constrained fog devices. Therefore, it is challenging for energy-aware security cognitive computing. One way is to design low complexity cryptographic algorithms, so that the encryption overhead and energy consumption of fog devices can be reduced. Another solution is to study energy-harvesting schemes for green IoV.

### B. Vehicular Fog Computing

DRL based solutions, relying on mass data for training or tons of documents for processing, are power consuming. A centralized cloud center is capable to obtain the overall network knowledge and can guarantee network security by efficient network management. However, cloud or GPU based deep learning may not cope with the rapid traffic demands in IoV. Therefore, Vehicular Fog Computing (VFC) can be integrated into the cloud computing model to enhance learning efficiency.

VFC is promising for real-time traffic management by making full use of idle resources in vehicles, such as vehicles on the move or in the parking slot [15]. However, the computational and storage capacities of VFC are still limited comparing with the cloud or cloudlet computing. In addition, the fog nodes generally lack resources to become intelligent by self-learning. In VFC, accurate prediction of vehicle mobility largely impacts the utilization of computing resources and energy. One way is to mine traffic flow according to the position, direction and velocity of vehicles. As EVs are becoming popular, vehicle-to-grid technology has been investigated to charge EVs and monitor their power status in the smart grid. A hybrid computing model for vehicle-to-grid networks is designed in [16], including a permanent cloud or cloudlet and temporary vehicular fog nodes. In [17], the authors design a hierarchical architecture enabled by VFC to provide prompt responses at neighborhood, community and city-wide traffic management, and detect events threatening network safety. Recently, L. Tan *et al* [18] design a mobility-aware edge caching vehicular networks. A classical machine learning method is developed to solve the advanced particle swarm optimization problem, which also guarantees the global optimal solution with fast convergence and high stability.

### C. Deep Reinforcement Learning based Offloading

Since the limited training data and novel applications appear continually, supervised learning becomes difficult for feature learning. Although unsupervised learning is promising to exploit the features of network traffic, it is challenging to fulfill real-time training processing [19]. The authors in [20] present an online reinforcement learning method to balance traffic loads in vehicular networks. By learning from the feedback and traffic patterns under dynamic vehicular circumstances, this method can well regulate network traffic. Because vehicles are not competent enough to analyze data or recognize traffic patterns in IoV, a DRL based framework is constructed to customize network services [21]. Due to the insufficient resource of fog devices, the realization of a complex deep learning system is challenging, not to mention the real-time deep learning for IoV with strict delay limitation.

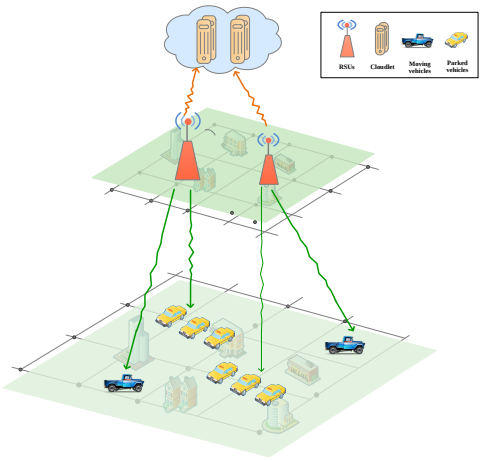



Fig. 1. Three-layer system model.

In order to fulfill high-efficient traffic management in IoV, a joint communication, caching and computing problem is investigated in [22]. A DRL based method is further presented to solve the mentioned problem. The authors in [23] propose a knowledge driven service offloading decision framework for IoV. They consider a long-term offloading scheduling, and provide the multi-task offloading policy by exploring the DRL based method. The resource allocation strategy is formulated as a joint optimization problem in [24], where the gains of not only networking but also caching and computing are taken into consideration. Due to the complexity of the formulated problem, a DRL based approach is developed to obtain the optimal solution. The authors in [25] transform the original joint computation offloading and content caching into a convex problem, and then solve it in a distributed and efficient way. Different from current researches that focus on content caching or distributed offloading schemes, our work presents an energy-efficient and load balancing offloading algorithm based on queuing theory.

## III. System Model

This section first provides an overview of the constructed three-layer system model. Then, the communication and computation models in the cloudlet and fog layers are specified, respectively. After that, the optimization problem focusing on the overall energy minimization is formulated.

### A. System Model

Fig. 1 illustrates the system model, including the layers of cloudlet, RodeSide Units (RSUs), and fog nodes. In our system, vehicles can send their offloading tasks to nearby RSUs. Instead of uploading all tasks to remote cloud servers or base stations, RSUs allocate these tasks to the cloudlet or nearby fog nodes for processing. Both parked

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2019.2930521, IEEE Transactions on Cognitive Communications and Networking



and moving vehicles can be viewed as fog nodes [15], and they are referred to be parked vehicle-based and moving vehicle-based fog nodes, respectively. Since computation offloading consumes massive electric power, our main focus is to minimize the overall energy consumption during the offloading procedure.

Through real dataset analysis, vehicular flows arriving at RSU $r_i$ follow a Poisson process, with an arrival rate denoted as $\lambda_i^{vehicle}$ [26]. Therefore, the uploading procedure of offloading tasks can be viewed as a subprocess of the vehicular flows. Correspondingly, the task flow arriving at RSU $r_i$ also follows a Poisson process, with an arrival rate $\lambda_i$. In addition, vehicular networks within urban city can be partitioned into several regions, and we focus on one of them as an example. The constructed offloading system is easy to be extended to other city-wide vehicular networks. In region $G$, there exists a cloudlet server $c$, a set of RSUs $\{r_1, \cdots, r_u\}$, a group of parked vehicles $\{v_1^p, \cdots, v_l^p\}$, and a stream of moving vehicles $\{v_1^m, \cdots, v_n^m\}$.

The objective of our work is to minimize the overall energy consumption of the offloading processes while satisfying the corresponding delay constraint. The energy consumption can be computed by $E = P \times t_{tol}$, where $P$ denotes the power of processing servers and $t_{tol}$ denotes the duration of the computation offloading. Specifically, the total execution delay of offloading can be obtained by $t_{tol} = t_{up} + t_{wat} + t_{pro} + t_{down}$, where $t_{up}$ and $t_{down}$ are transmission time of uploading and downloading between an RSU and a process server. The symbol $t_{wat}$ denotes the queue waiting time for a task at the process server, and the processing time is represented by $t_{pro}$. For simplicity, $t_{up}$ can be regarded as equal to $t_{dowm}$. Therefore, $t_{tol} = 2t_{up} + t_{wat} + t_{pro}$ denotes system execution delay for offloading.

### B. Cloudlet Model

Since the computation capability of vehicles are limited, cloudlet can be leveraged to process offloading tasks with strict delay requirement. The total execution time contains four parts: $t_{tol}^c = t_{up}^c + t_{wat}^c + t_{pro}^c + t_{down}^c$. In addition, the cloudlet processing system can be modeled as an $M/M/b$ queue, including $b$ homogeneous servers with fixed processing rate $\mu_c$. Let $\lambda^c$ denote the number of offloading flow waiting in the queue. The service rate for cloudlet can be computed by $\rho^c = \lambda^c / b\mu_c$, and $\rho^c < 1$ holds. According to queuing theory [27], expectation waiting time in queue $\mathbb{E}(t_{wat})$ for a task can be computed by:

$$\begin{aligned}\mathbb{E}(t_{wat}) &= f(b, \rho) \\ &= \left[\sum_{k=0}^{b-1} \frac{\left(\frac{b}{k}\right)!\left(1-\rho^2\right)}{(b\rho)^{b-k}\rho} + \frac{1-\rho}{\rho}\right]^{-1}.\end{aligned} \tag{1}$$

The expectation processing time is $\mathbb{E}(t_{pro}^c) = 1/\mu_c$. In addition, the uploading time $t_{up}$ is related to achievable transmission rate, which can be calculated by:

$$R_{r_i \to cloudlet} = W \log_2\left(1 + \frac{p_{i,c} h_{i,c}}{\sigma^2 + \sum_{j=1, j\neq i}^{u} p_{j,c} h_{j,c}}\right), \tag{2}$$

where $W$ is the allocated bandwidth. Transmission power $p_{r_i \to cloudlet}$ and channel gain $h_{r_i \to cloudlet}$ are simplified as $p_{i,c}$ and $h_{i,c}$, respectively. Variable $\sigma^2$ denotes the Guassian noise. Non-Orthogonal Multiple Access (NOMA) technology is utilized for the connection between cloudlets and RSUs. The interference caused by other RSUs occupying the same channel is represented by $\sum_{j=1, j\neq i}^{u} p_{j,c} h_{j,c}$.

Vehicles are considered to generate offloading tasks $T = \{d, t_{max}\}$, where $d$ represents the data size to be processed, and $t_{max}$ denotes the maximal tolerable delay. Therefore, the uploading time can be calculated by $t_{up}^c = d_{r_i \to cloudlet} / R_{r_i \to cloudlet}$. The expectation of total execution time $t_{tol}^c$ can be obtained by:

$$\begin{aligned}\mathbb{E}(t_{tol}^c) &= 2 \times t_{up}^c + \mathbb{E}(t_{wat}^c) + \mathbb{E}(t_{pro}^c) \\ &= 2 \times \frac{d_{r_i \to cloudlet}}{R_{r_i \to cloudlet}} + f(b, \rho^c) + \frac{1}{\mu_c}.\end{aligned} \tag{3}$$

Thus, the total energy consumption of cloudlet offloading procedure can be calculated by:

$$\begin{aligned}E_{tol}^c &= E_{up}^c + E_{wat}^c + E_{pro}^c + E_{down}^c \\ &= (P_{i,c}^{up} + P_{c,i}^{dowm}) t_{up}^c + P_c^{wat} \mathbb{E}(t_{wat}^c) + P_c^{pro} \mathbb{E}(t_{pro}^c).\end{aligned} \tag{4}$$

### C. Fog Model

As mentioned in Section III-A, both parked and moving vehicles can be leveraged for fog nodes. In this subsection, we specify the fog model.

*1) Parked Vehicle-based Fog Model:* Considering that there are $s$ time slots in a day, the number of parked vehicles $l$ keeps unchanged during each time slot. A parked vehicle can be regarded as a fog node equipped with one processing server (with a fixed service rate $\mu_p$). These $l$ parked vehicle based fog nodes can be modeled as an $M/M/L$ queue. Similar to the cloudlet, the service rate of the parked vehicles can be calculated by $\rho_i^p = \lambda_i^p / l\mu_p$, where $\lambda_i^p$ represents the flow waiting number for parked vehicles to process, and $\rho_i^p < 1$ holds. The expectation of total execution time for one task can be obtained by:

$$\begin{aligned}\mathbb{E}(t_{tol}^p) &= \mathbb{E}(t_{wat}^p) + \mathbb{E}\left(t_{pro}^p\right) + 2 \times t_{up}^p \\ &= f\left(l, \rho_i^p\right) + \frac{1}{\mu_p} + 2 \times t_{up}^p.\end{aligned} \tag{5}$$

Then, the total energy consumption of parked vehicles can be computed by:

$$\begin{aligned}E_{tol}^p &= E_{up}^p + E_{wat}^p + E_{pro}^p + E_{down}^p \\ &= (P_{i,p}^{up} + P_{p,i}^{dowm}) t_{up}^p + P_p^{wat} \mathbb{E}(t_{wat}^p) + P_p^{pro} \mathbb{E}(t_{pro}^p).\end{aligned} \tag{6}$$

*2) Moving Vehicle-based Fog Model:* A moving vehicle can also be viewed as a fog node with a processing server. Due to its mobility, it is more complicated than a parked vehicle. It is demonstrated that moving vehicles can be modeled as an $M/M/1$ queue with an arrival rate $\lambda_i^{vehicle}$ [15].

Since RSU $r_i$ allocates the arriving offloading flow to cloudlet, parked vehicles and moving vehicles, offloading flows arriving at moving vehicles $\lambda_i^m$ can be viewed as a subpart of the offloading flow arriving at RSU $r_i$, with arriving

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2019.2930521, IEEE Transactions on Cognitive Communications and Networking



rate $\lambda_i$. Thus, $\lambda_i^m < \lambda_i$ holds. Correspondingly, $\lambda_i^m < \lambda_i < \lambda_i^{vehicle}$ holds. In addition, we assume that the computation capabilities of all moving vehicles are the same for simplicity. When a moving vehicle enters in the wireless communication range of RSU $r_i$, it picks up an offloading task waiting in front of the queue, and can accomplish the processing before moving out of the communication range. The offloading flow waiting in the queue can be regarded as a stochastic process, represented as $\{X_n, n \geq 0\}$. The discrete status, $i_0, i_1, ..., i_{n+1}$, indicates the number of tasks in the waiting queue, and $P\{X_0 = i_0, X_1 = i_1, ..., X_n = i_n\} > 0$ holds. It can be observed that the number of offloading tasks in the waiting queue at time slot $t+1$ only depends on the status at time slot $t$, i.e., $P\{X_{n+1} = i_{n+1} \mid X_0 = i_0, ..., X_n = i_n\} = P\{X_{n+1} \mid X_n = i_n\}$. Therefore, the offloading flow waiting for moving vehicles to process can be modeled as a Markov chain. The transition rate matrix is as follows:

$$\begin{pmatrix} -\lambda_i^m & \lambda_i^m & & \\ \lambda_i^{vehicle} & -\left(\lambda_i^{vehicle} + \lambda_i^m\right) & \lambda_i^m & \cdots \\ & \lambda_i^{vehicle} & -\left(\lambda_i^{vehicle} + \lambda_i^m\right) & \\ & & \lambda_i^{vehicle} & \\ \cdots & & & \end{pmatrix}. \tag{7}$$

We can observe that the above transition rate matrix is formally identical to the $M/M/1$ queuing system. Therefore, the offloading system of moving vehicle based fog nodes can be modeled as an $M/M/1$ queue, where $\lambda_i^m$ denotes the task arrival rate, and the processing rate of moving vehicles is $\lambda_i^{vehicle}$. Thus, the service rate can be calculated by $\rho_i^m = \lambda_i^m / \lambda_i^{vehicle}$. In addition, the expectation of total execution time can be computed by:

$$\begin{aligned} \mathbb{E}(t_{tol}^m) &= \mathbb{E}(t_{wat}^m) + \mathbb{E}\left(t_{pro}^m\right) + 2 \times t_{up}^m \\ &= \frac{\rho_i^m}{\lambda_i^{vehicle}\left(1 - \rho_i^m\right)} + \frac{1}{\lambda_i^{vehicle}} \\ &\quad + 2 \times t_{up}^m. \\ &= \frac{1}{\lambda_i^{vehicle} - \lambda_i^m} + 2 \times t_{up}^m. \end{aligned} \tag{8}$$

Correspondingly, the total energy consumption of moving vehicles can be obtained by:

$$\begin{aligned} E_{tol}^m &= E_{up}^m + E_{wat}^m + E_{pro}^m + E_{down}^m \\ &= (P_{i,m}^{up} + P_{m,i}^{dowm}) t_{up}^m + P_m^{wat} \mathbb{E}(t_{wat}^m) + P_m^{pro} \mathbb{E}(t_{pro}^m). \end{aligned} \tag{9}$$

### D. Redirection Model

In order to balance offloading task flows, RSUs are considered to be reachable from each other, similar to [28]. Since flows arriving at RSUs may be significantly different, overloaded RSU can redirect part of its flows to unloaded RSUs. Variable $g(i,k)$ is defined as the amount of task flows redirected from RSUs $r_i$ to $r_k$. In addition, the following constraints should be satisfied:

$$g(i,k) = \begin{cases} -g(k,i), & i \neq k, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

$$\sum_{i=1}^{u} \sum_{k=1}^{u} g(i,k) = 0, \tag{11}$$

$$\sum_{k=1}^{u} \max\{g(i,k), 0\} \leq \lambda_i, \tag{12}$$

where indicators $i, k \in \{1, 2, ..., u\}$. The communication channel states are assumed to be identical and keep unchanged during one scheduling time slot. Let $d_{r_i, r_k}$ indicate unit transmission delay caused by transferring offloading tasks between RSU $r_i$ and RSU $r_k$. If redirected flow $g(i,k) < 0$, RSU $r_k$ transfers flows to RSU $r_i$ and delay $(-g(i,k) \times d_{r_i, r_k})$ exists. The total delay $t_{i,redirect}^j$ incurred by redirecting flows from RSU $i$ to other RSUs during time slot $j$ can be calculated as follows:

$$t_{i,redirect}^j = \sum_{k=1}^{u} |\max\{g(i,k), 0\} \times d_{r_i, r_k}|. \tag{13}$$

In addition, the energy consumption of redirection for RSU $r_i$ at time slot $j$ can be obtained by $E_{i,redirect}^j = P_{r_i} \times t_{i,redirect}^j$. The final arriving task flows at RSU $r_i$ can be computed by:

$$\overline{\lambda}_i = \lambda_i - \sum_{i=1}^{u} g(i,k). \tag{14}$$

## IV. Problem Formulation

The optimization problem is formulated in this section. Since the formulated problem is a Mixed Integer Non-Linear Programming (MINLP) problem, and variables are tightly coupled in different constraints, the optimization problem is divided into two sub-problems.

### A. Optimization Objective

When a vehicle generates an offloading task, it uploads the task flow to the nearby RSU. After RSU receives accurate information of all tasks, it will judge whether it is overloaded or not. Then, RSUs perform flow redirection to achieve load balance among them. In order to minimize energy consumption, RSUs make offloading decisions on the redirected task flow.

In our work, urban area is divided into several regions. In each region, a central cloudlet and $u$ distributed RSUs are available. Both parked and moving vehicles in the proximity of RSUs can be utilized as fog nodes. The total energy consumption of RSU $r_i$ at time slot $j$ can be calculated by:

$$E_{i,tol}^j = \alpha E_{i,tol}^{c(j)} + \beta E_{i,tol}^{p(j)} + \gamma E_{i,tol}^{m(j)} + E_{i,redirect}^j, \tag{15}$$

where $\alpha + \beta + \gamma = 1$, and

$$\alpha = \begin{cases} 1, & \text{if the message is processed by the cloudlet,} \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

$$\beta = \begin{cases} 1, & \text{if the message is processed by the parked} \\ & \text{vehicle based fog nodes,} \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2019.2930521, IEEE Transactions on Cognitive Communications and Networking



$$\gamma = \begin{cases} 1, & \text{if the message is processed by the moving} \\ & \text{vehicle based fog nodes,} \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

Therefore, the optimization problem is defined as: if network parameters $(G, \mu_c, \mu_p, d_{r_i \to cloudlet}, \lambda_1, ...\lambda_s, l,\ \lambda_i^{vehicle})$ can be obtained, the optimization target is to find available $\lambda_i^c$, $\lambda_i^p$, $\lambda_i^m$ and $b$, such that the average energy consumption at each time slot can be minimized, i.e.,

$$\min_{\lambda_i^c, \lambda_i^p, \lambda_i^m, b} \frac{1}{su} \sum_{j=1}^{s} \sum_{i=1}^{u} E_{i,tol}^j, \tag{19}$$

$$\textbf{s.t.} \begin{cases} Equations\ (10) - (14), \\ \lambda_i^c + \lambda_i^p + \lambda_i^m \leq \overline{\lambda}_i, \\ \lambda^c = \sum_{i=1}^{u} \lambda_i^c, \\ 0 < \frac{\lambda^c}{b\mu_c}, \frac{\lambda_i^p}{l\mu_p}, \frac{\lambda_i^m}{\lambda_i^{vehicle}} < 1, \\ \alpha + \beta + \gamma = 1, \alpha, \beta, \gamma \in \{0, 1\}. \end{cases} \tag{20}$$

Based on above constraints, we can observe that the formulated problem is a MINLP problem, which is difficult to solve. Thus, the original problem is divided into two solvable sub-problems.

### B. Flow Redirection

Distributed in various urban areas, the amount of task flows arriving at RSUs may be significantly distinct. If RSUs only process tasks within their regions, RSUs located at business district or stations may be overloaded. Therefore, load balancing is necessary before offloading decision making. Since cloudlet server is generally equipped with sufficient electric energy, the optimization target of load balancing is to balance the processing load among fog nodes (i.e., parked vehicles and moving vehicles) through flow redirection.

Load ratio $\sigma$ is defined as the ratio of processing rate by fog nodes compared to arriving task flows at RSUs, i.e.,

$$\sigma = \frac{\sum_{i=1}^{u} (l\mu_p + \lambda_i^{vehicle})}{\sum_{i=1}^{u} \lambda_i}. \tag{21}$$

RSUs are divided into two sets, where one is unloaded set $V_u = \left\{k | l\mu_p + \lambda_k^{vehicle} \leq \lambda_k\right\}$, and the other is overloaded set $V_o = \left\{i | l\mu_p + \lambda_i^{vehicle} > \lambda_i\right\}$.

For each overloaded RSU $r_i, i \in V_o$, the amount of task flows that can be redirected to other RSUs is defined as $\phi_i$. Based on Equation (21), $\phi_i$ can be obtained by $\phi_i = \lambda_i \times \sigma - l\mu_p - \lambda_i^{vehicle}$, where $\phi_i > 0$ holds. Similarly, for each unloaded RSU, the amount of task flows allowing to be redirected from other RSUs can be calculated by $\phi_k = l\mu_p + \lambda_k^{vehicle} - \lambda_k \times \sigma$, where $\phi_k > 0$ holds. Then, the first sub-problem is formulated as follows:

$$\min \frac{1}{s} \sum_{j=1}^{s} \sum_{i \in V_o} \sum_{k \in V_u} E_{i,redirect}^j. \tag{22}$$

The objective is to minimize the energy consumption of redirection for all time slots. Since flow redirection is identical for each time slot, it is equivalent to minimize the energy consumption in each time slot, where the objective is:

$$\min \sum_{i \in V_o} \sum_{k \in V_u} E_{i,redirect}. \tag{23}$$

$$\textbf{s.t.} \begin{cases} Equations\ (10) - (14), \\ \sum_{i \in V_o} g(i,k) = \phi_k, k \in V_u, \\ \sum_{k \in V_u} g(i,k) = \phi_i, i \in V_o, \\ g(i,k) \geq 0. \end{cases} \tag{24}$$

### C. Offloading Decision

After redirecting task flows, RSUs make offloading decisions by scheduling arrived task flows to cloudlet or fog nodes. To minimize the average energy consumption of task offloading, the objective of the second sub-problem can be expressed as follows:

$$\min_{\lambda_i^c, \lambda_i^p, \lambda_i^m, b} \frac{1}{su} \sum_{j=1}^{s} \sum_{i=1}^{u} (\alpha E_{i,tol}^{c(j)} + \beta E_{i,tol}^{p(j)} + \gamma E_{i,tol}^{m(j)}). \tag{25}$$

After redirecting task flows, the optimization objective, i.e., minimizing the average energy consumption for all RSUs, is equivalent to minimizing the energy consumption for each RSU. Therefore, the second sub-problem becomes:

$$\min_{\lambda_i^c, \lambda_i^p, \lambda_i^m, b} \frac{1}{s} \sum_{j=1}^{s} (\alpha E_{i,tol}^{c(j)} + \beta E_{i,tol}^{p(j)} + \gamma E_{i,tol}^{m(j)}), \tag{26}$$

$$\textbf{s.t.} \begin{cases} \lambda_i^c + \lambda_i^p + \lambda_i^m \leq \overline{\lambda}_i, \\ \lambda^c = \sum_{i=1}^{u} \lambda_i^c, \\ 0 < \frac{\lambda^c}{b\mu_c}, \frac{\lambda_i^p}{l\mu_p}, \frac{\lambda_i^m}{\lambda_i^{vehicle}} < 1, \\ \alpha + \beta + \gamma = 1, \alpha, \beta, \gamma \in \{0, 1\}. \end{cases} \tag{27}$$

## V. Overview of Deep Reinforcement Learning

Before implementing the DRL based algorithm, we briefly provide an overview of DRL.

Reinforcement Learning (RL) is a significant branch of machine learning. It refers to the process of achieving a target through multiple steps and appropriate decisions under a series of scenarios, which can be regarded as a multi-step sequence decision problem. Different from traditional machine learning, RL can not obtain final results immediately, but only observes a temporary reward (mostly set by human experience). Thus, RL can also be viewed as a kind of delayed supervision learning.

Dynamic Programming (DP) and policy-based optimization algorithms are generally leveraged to solve the problems in RL, where DP includes value iteration and policy iteration. In

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2019.2930521, IEEE Transactions on Cognitive Communications and Networking



model-free algorithms, DP can be classified as Monte Carlo and temporal difference algorithms. With the development of RL, it is utilized to deal with problems with continuous state and action spaces. It is almost impossible to measure each state or action with a simple value function. Therefore, policy-based optimization is proposed by parameterizing the policy, which introduces neural networks into RL.

Q-learning is a typical temporal difference RL algorithm. It defines Q-function to evaluate the long-term reward of the policy, which is replaced by neural networks in DRL. For each episode, Q-learning makes decisions based on the Q-value, which evaluates the selected action under current circumstances. Deep learning fits data distribution and function model through multi-layer nonlinear neural networks. DRL integrates the superiority of both RL and deep learning. In addition, based on the characteristics of trial and error and reward delay, the performance of DRL is satisfactory in realistic decision making problems.

Experience replay can improve the utilization ratio of samples while reducing the correlation. Random sampling means that each transition is sampled with the same probability. However, the value of all samples may vary greatly. Learning simple and common samples does not improve the Deep Q-Network (DQN) model significantly. If DQN treats all transitions equally, it can spend much time on ordinary samples, which can not fully exploit the potential of the training data. Priority experience replay solves this deficiency by assigning a certain sampling weight according to the performance of the current sample. The worse the performance is, the higher the weight is assigned, resulting in the higher the probability of sampling becomes. In this way, those samples, decreasing the model performance, have high probabilities to be re-learned.

In the training phase, every step of sampling is random. To obtain accurate Q-values, we can perform multiple samples to obtain the expected value of Q-network, which is the key idea of the asynchronous DRL. Thus, the equation for reward calculation becomes:

$$Q = \mathbb{E}(r + \gamma \mathcal{Q}(s', arg\max_{a'} \mathcal{Q}(s', a'; \theta); \theta^{-})). \tag{28}$$

In practical applications, asynchronous solutions are usually performed in multiple threads.

## VI. Deep Reinforcement Learning Based Offloading Algorithm

In this section, Edmonds-Karp algorithm and DRL based schemes are leveraged to solve the formulated two sub-problems, respectively.

### A. Flow Redirection

An example of flow redirection is illustrated in Fig. 2, where overloaded RSU $r_i$ redirects part of its received offloading task flows to unloaded RSU $r_k$. The first sub-problem (i.e., Equation (23)) can be viewed as a typical network flow problem, where RSUs and energy consumption are network nodes and cost, respectively. In addition, the topology of RSUs

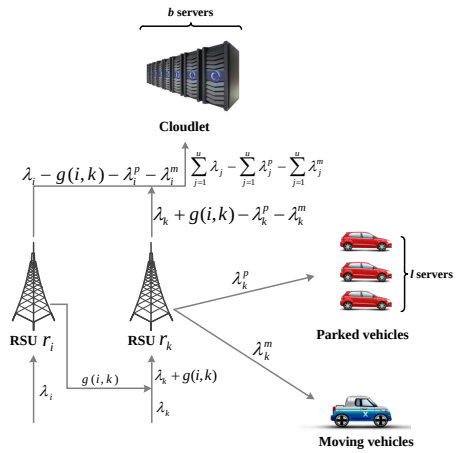



Fig. 2. Flow redirection from RSUs $r_i$ to $r_k$.

within $G$ constitutes a network flow graph.

Edmonds-Karp algorithm based scheme is leveraged to solve this problem, and its pseudocode is specified in Algorithm 1. In order to construct a maximum flow problem, a super source node $s$ and a super destination node $t$ are added in graph $G$ to form $G^{'}$, where RSUs in $V_o$ and $V_u$ connect to nodes $s$ and $t$, respectively. For RSUs in $V_o$, the capacity of the connection between RSU $r_i$ and source node $s$ is $\phi_i$. Similarly, the capacity of the connection between RSU $r_k \in V_u$ and destination node $t$ is $\phi_k$. Note that the transmission delay among RSUs and super nodes is 0. By the breadth-first search method, Edmonds-Karp algorithm based scheme searches the shortest augmentation path in $G^{'}$ until no more augmentation path exists. In particular, it can fulfill Ford-Fulkerson algorithm with polynomial time complexity. Flow redirection is promising to make more RSUs not overloaded under the premise of minimum energy consumption.

### B. DRL for Energy Consumption Minimization

Different from traditional machine learning algorithms, DRL does not prepare training data in advance, but samples unlabeled data. Traditional DQN utilizes the same Q-network to evaluate the selection of system actions and approximate the true action-value function. Although this method improves the robustness of the DQN model, action selection by the greedy strategy results in inherent shortcomings of DQN, e.g., overestimation. Therefore, Double DQN (DDQN) is proposed in [29] by constructing different action-value functions to evaluate the selection scheme (i.e., the evaluated Q-network) and approximate the real action-value function (i.e., the target Q-network). When implementing DDQN, DQN copies the parameter of evaluated Q-network as the target Q-network. Then, evaluated Q-network and target Q-network update alternately. The proposed DRL based algorithm for Minimizing Energy Consumption (DMEC) is based on DDQN. Note that implementing a simple Q-learning algorithm through neural networks in DRL model can cause shocks or divergence. The reasons are follows:

- Data serialization: Time-continuous samples are interrelated and not independently distributed;

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2019.2930521, IEEE Transactions on Cognitive Communications and Networking



**Algorithm 1:** Pseudo-code of Edmonds-Karp algorithm based delay minimization scheme

**Input:** Graph of RSUs $G$, $V_u$, $V_o$ $delay$, $capacity$
**Output:** $flows$, $minEnergyConsumption$
Add a super source node $s$ and a super destination node $t$ in $G$ to form $G^{'}$;
**while** $i \in V_o$ **do**
  $capacity[s][i] = \phi_i$;
  $delay[s][i] = 0$;
**end**
**while** $k \in V_u$ **do**
  $capacity[k][t] = \phi_k$;
  $delay[k][t] = 0$;
**end**
**for** *each edge* $e[i][j]$ *in* $G^{'}$ **do**
  $flows[i][j] = 0$;
**end**
**while** *find a shortest route* $p$ *from* $s$ *to* $t$ *in* $G^{'}$ **do**
  $m = min(capacity[i][j]|e[i][j] \in p)$;
  **for** *each edge* $e[i][j]$ *in* $p$ **do**
    **if** $m <= capacity[i][j]$ **then**
      $capacity[j][i] = capacity[j][i] + m$;
      $flows[i][j] = flow[i][j] + m$;
    **end**
  **end**
  $minEnergyConsumption = P_{r_i} \times flows \times delay$;
**end**

- Policy may oscillate: Minor changes in Q values may dramatically affect policies;
- The value range of reward functions is unknown: Gradients in simple Q-learning are very unstable when back-propagating.

In order to overcome the mentioned obstacles, experience replay is investigated to prevent training results from falling into local optimum. In addition, it utilizes random sampling to simulate supervised learning and break the correlation among data training. Specific steps of experience replay are: initialize a buffer in memory, and store Markov Decision Process (MDP) based training data during simulation. Then, randomly sample a batch of samples in stored training data. After that, calculate Q value and update network parameters. When the experience replay buffer is full, the newly sampled training data substitute the old data. The exploition of the experience replay mechanism can be viewed as the utilization of Monte Carlo algorithm in deep Q-learning.

The second sub-problem (i.e., Equation (25)) is a cost minimization problem with a period of time. Due to the immense space of available offloading decisions and time-varying characteristics, the energy consumption minimization problem can be formulated as a DRL process. The DRL problem can be viewed as an optimal control problem based on MDP, where convolutional neural networks are leveraged to represent the action-value function. Specifically, there are four significant elements in DRL: agents, system states, system actions and rewards. In the following, they are identified to construct the DRL-based model:

- Agents: RSUs are selected to be agents in the DRL process. For each episode, RSUs choose an action according to system states, with the target of maximizing rewards.
- System States: They comprise the arrival rate $\lambda_i^{vehicle}$ that moving vehicular flows arriving at RSU $r_i$, the arrival rate of offloading task flows $\lambda_i$, and the number of parked vehicles within the communication range of RSU $l$. Since the current values of system states are only related to the value of the last time slot, they are modeled as Finite-State Markov Chains (FSMC) [30], which can be expressed as:
$$\mathcal{S}_i(t) = [\lambda_i^{vehicle}(t), \lambda_i(t), l(t)]. \tag{29}$$
- System Actions: RSU $r_i$ is responsible for scheduling arrival task flow $\lambda_i$ to three platforms: cloudlet $\lambda_i^c$, parked vehicles $\lambda_i^p$, and moving vehicles $\lambda_i^m$. The infrastructure based cloudlet generally consumes more energy than that of fog nodes. Thus, task flows are preferentially assigned to parked vehicle based and moving vehicle based fog nodes. In addition, moving vehicles are not stable, and their locations vary a lot during a time slot. Therefore, task flows assigned to parked vehicles satisfy $\lambda_i^p = l\mu_p$. Then, RSUs decide the value of $\lambda_i^m$ through DRL. Finally, the rest task flows are scheduled to the cloudlet: $\lambda_i^c = \lambda_i - \lambda_i^p - \lambda_i^m$. Since the value of the task flow is continuous in reality, the available system action space is infinite. Therefore, the value range of $\lambda_i^m$ should be discretized and quantized into $L = \min\left\{\lambda_i^{vehicle}, |\lambda_i - \lambda_i^p|\right\}$ levels, which is presented as:
$$a_i(t) = [0, 1, 2, \cdots, L-1, L]. \tag{30}$$
- Rewards: Generally, agents aim to maximize rewards for all time slots in DRL. However, the optimization target of our model is to minimize the overall energy consumption. Thus, the reward function is defined as the opposite number of the energy consumption in Equation (23), which is expressed as follows:
$$\mathcal{R}_i = -\frac{1}{s}\sum_{j=1}^{s} \alpha E_{i,tol}^{c(j)} + \beta E_{i,tol}^{p(j)} + \gamma E_{i,tol}^{m(j)} \tag{31}$$

The pseudo-code of DMEC is presented in Algorithm 2. First, experience replay buffer and parameters of DQN are initialized. Specially, the evaluated Q-network is initialized with the same parameter to the target Q-network, i.e., $\theta = \theta^-$.

For each episode, RSUs choose actions randomly with probability $\varepsilon$. A greedy strategy is leveraged to select the action that maximizes current rewards. Then, the immediate reward and the next system states can be observed. In addition, current system states, the selected action, the obtained rewards and the next observed system states constitute a transition, which is stored in the replay buffer to train target Q-network. In the training phase, a mini-batch transition is randomly sampled from replay buffer to eliminate the coupling of time series training data. For each transition, if the next state is the termination state, the immediate reward is the temporal difference target of training. Otherwise, target Q-network is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2019.2930521, IEEE Transactions on Cognitive Communications and Networking



**Algorithm 2:** Pseudo-code of DMEC.

**Input:** system states $\mathcal{S}_i(t)$, available action space $a_i(t)$
**Output:** maximum rewards $\mathcal{R}_i$
Initialize the experience replay buffer;
Initialize the evaluated Deep Q-Network with weights $\theta$;
Initialize the target Deep Q-Network with weights $\theta^- = \theta$;
**for** *each episode* $i = 1, 2, \cdots, U$ **do**
  Initialize observation $s_1$, and pre-process sequence $x_1 = \varphi(s_1)$;
  **for** $t = 1, 2, \cdots, T-1$ **do**
    With probability $\varepsilon$ select a random action $a_{i,k}(t)$ Otherwise, select $a_t = arg\max_{a_{i,k}(t)} \mathcal{Q}(x, a; \theta)$;
    Execute action $a_{i,k}(t)$;
    Observe the immediate reward $r_t = \mathcal{R}_i(t)$ and the next observation $s_{t+1}$;
    Process $s_{t+1}$ to be the next state $x_{t+1} = \varphi(s_{t+1})$;
    Store transition $(x_t, a_t, r_t, x_{t+1})$ into $D$;
    Sample random mini-batch of transitions $(x_j, a_i, r_j, x_{j+1})$ from $D$;
    **if** *episode terminates at step* $j+1$ **then**
      the target $\mathcal{Q}$-value $y_j = r_j$;
    **end**
    **else**
      $y_j = r_j + \gamma \mathcal{Q}(x_{j+1}, arg\max_{a'} \mathcal{Q}(x_{j+1}, a'; \theta); \theta^-)$;
    **end**
    Perform gradient decent on $\Big(y_j - \mathcal{Q}(x_j, a_j; \theta)\Big)^2$;
    Every $C$ steps, update the target Deep Q-Network parameters and probability $\varepsilon$ with rate $\sigma$ and $\mu$,

$$\theta^- = \sigma\theta + (1-\sigma)\theta^-,$$

$$\varepsilon = \varepsilon - \mu\varepsilon.$$

  **end**
**end**

leveraged to calculate the temporal difference target. After that, gradient descent is performed to update parameters of the evaluated Q-network. Finally, by considering both fitting accuracy and convergence speed, parameters of target Q-network and random probability $\varepsilon$ are updated every $C$ steps, where $C$ is a constant.

### C. Complexity Analysis

In this subsection, we theoretically analyze the complexity of our proposed two algorithms. For the Edmonds-Karp algorithm based flow redirection, we assume that all RSUs can communicate with each other, i.e., the flow redirection graph is a complete graph. Therefore, the computation complexity of the Edmonds-Karp algorithm based scheme is $O(u^3)$, where $u$ represents the number of RSUs. Since vehicles do not participate in the flow redirection, the complexity of the Edmonds-Karp algorithm based scheme does not scale when the number of vehicles increases. Since the number of RSUs is much less than that of vehicles, its computation complexity is acceptable.

With neural networks, the computation complexity of the DMEC algorithm is related to the number of training and updating parameters. Let $U$ denote the total training episodes, which can be viewed as a hyper-parameter. For each episode, the DMEC loops until the offloading task is terminated, where the total number of time slots is denoted as $T$. Without loss of generality, the computation complexity of gradient decent and parameter updating can be defined as $M$ and $N$, respectively. In order to improve training efficiency, the parameter of target DQN is updated every $C$ steps, where $C$ is a constant. In summary, the computation complexity of the DMEC is $O(U \times T((M+N)/C)$. Note that the size of the input layer of the DMEC is the same as that of the system states, which is proportional to the number of vehicles. Thus, the growing number of vehicles can lead to the scaling of the computation complexity of the DMEC algorithm. Two methods are utilized to decrease the computation complexity. The first one is increasing $C$ to decline the computation complexity at the expense of training accuracy. The second one is limiting the number of vehicles by dividing a city into multiple districts according to the map or administrative areas. Performance evaluations based on real-world traces of taxies in Jingan district, Shanghai (China), demonstrate the effectiveness of our proposed method in the next section.

## VII. Peformance Evaluation

With python anaconda 4.3, TensorFlow 0.12 is employed to implement the DMEC algorithm on Ubuntu 16.04 LTS. To balance the trade-off between training accuracy and time consumption, we select Jingan district in Shanghai (China) as illustrated in Fig. 3, while the method can be extened to other districts. One cloudlet server and 5 RSUs equipped with MEC servers are considered to be within the district. The selected GPS locations are presented in Table I. Performances are evaluated based on real-world traces of taxies that is collected in the whole April 2015, including the GPS information of more than 1000 taxies. The learning rate of the DMEC is set as 0.99. There are 3 hidden layers (each with 128 full connected neurons) for both evaluated Q-Networks and target Q-Networks. The total training episodes are 20000, and the parameter of the target Q-Network is updated every 10 time slots. On the flow redirection stage, our optimization target is to minimum energy consumption under the premise of RSUs not overloaded. Parameter delay matrix can be calculated according to Equation (13). In addition, there are two performance indicators, i.e., average energy consumption of flow redirection and overloaded ratio of RSUs.

For the second stage, key parameters are illustrated in Table II, where $W^c$ and $W^v$ denote the bandwidth that RSUs communication with cloudlets and vehicles, respectively. Three schemes are selected to compare with the presented DMEC:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2019.2930521, IEEE Transactions on Cognitive Communications and Networking



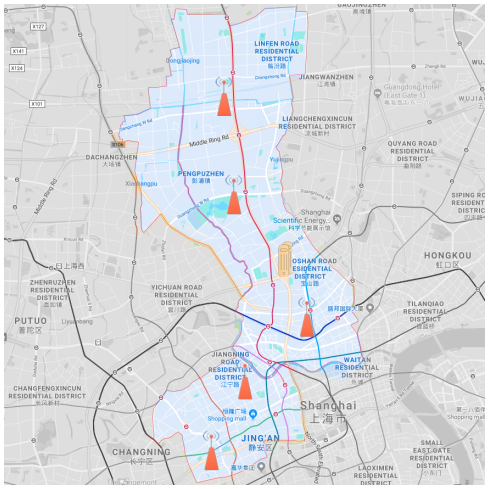



Fig. 3. Map of Jingan District in Shanghai (China).

TABLE I
SELECTED GPS LOCATIONS IN JINGAN DISTRICT

| Server ID | GPS locations |
|---|---|
| Cloudlet | 31.267186, 121.467802 |
| RSU 1 | 31.299528, 121.456591 |
| RSU 2 | 31.241008, 121.451848 |
| RSU 3 | 31.281631, 121.456447 |
| RSU 4 | 31.317300, 121.450698 |
| RSU 5 | 31.257432, 121.476282 |

TABLE II
SIMULATION PARAMETERS

| Simulation Parameter | Value |
|---|---|
| Number of RSUs $u$ | 5 |
| Bandwidth $W^c/W^v$ | 4/1 MHz |
| Processing capability of parked vehicles $\mu_c/\mu_p$ | 2-6 Messages/slot |
| Number of parked vehicles $l$ | 100-200 |
| Data size of offloading tasks $d$ | 20-100 MB |
| Offloading task arrival rates $\lambda_i$ | 200-400 tasks/slot |
| Noise power $\sigma$ | -114 dBm |

- Q-learning: It is a traditional temporal difference algorithm, which always pursues the largest reward at the next time slot. In addition, Q-learning records rewards in each iteration. When system state or action spaces are large, it can occupy massive memories.
- MEES [31]: It is a heuristic algorithm that aims at minimizing the energy consumption of MEC-enabled RSUs by jointly considering task scheduling among MEC servers and downlink energy consumption of RSUs.
- Cloudlet computing: All arriving offloading tasks are transferred to the cloudlet server for processing.

Fig. 4 illustrates the performance of average energy consumption of flow redirection with different offloading task arrival rates. We can observe that the average energy consumption raises gradually with the increasing of the offloading task arrival rate. It is because that increasing tasks lead to more overloaded RSUs, and more task flows need to be redirected. In addition, when the task arrival rate is relatively low (e.g., 15 or 20), little distinction exists among the performance of Edmonds-Karp, the greedy and the exhaustive algorithms.

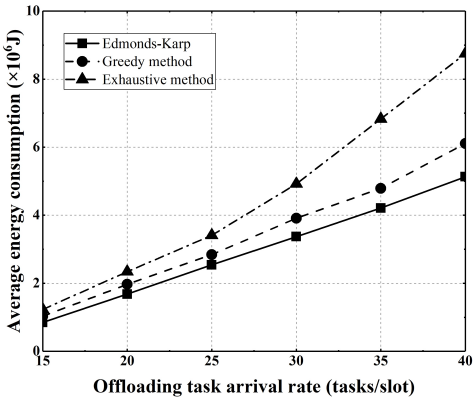



Fig. 4. Average energy consumption of flow redirection with different offloading task arrival rates.

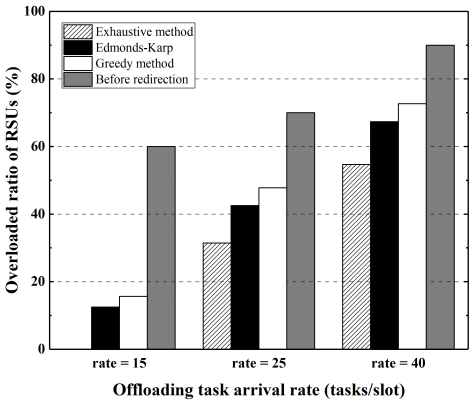



Fig. 5. Overloaded ratio of RSUs with different offloading task arrival rates.

As the arrival rate increases, the energy consumed by the exhaustive method raises sharply. This is because that the number of searching is small when the arrival rate is low, and grows quickly with the increase of the arrival rate. Edmonds-Karp algorithm based scheme always chooses the current shortest augmentation path in the network flow graph through breath-first searching. Therefore, it is superior to the greedy method.

Fig. 5 shows the performance of the overloaded ratio of RSUs with different offloading task arrival rates. Herein, the overloaded ratio of RSUs is defined as $|V_o|/(|V_u|+|V_o|)$. For ease of description and without loss of generation, we select three situations to illustrate the performance, where the task arrival rate equals to 15, 25 and 40, respectively. It can be observed that the performance of flow redirection based on low task arrival rate is better than that of based on high task arrival rate. It is because that when the arrival rate is high, more than half of RSUs are overloaded, and the space of flow redirection is limited. Thus, the overloaded ratio of RSUs still maintains high after flow redirection. In addition, even though exhaustive method performs better than the Edmonds-Karp algorithm based scheme, it consumes more energy at the same time. In summary, the Edmonds-Karp algorithm based scheme can achieve effective and energy efficient performances.

Fig. 6 illustrates the performance of average energy consumption with different offloading task arrival rates for one

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2019.2930521, IEEE Transactions on Cognitive Communications and Networking



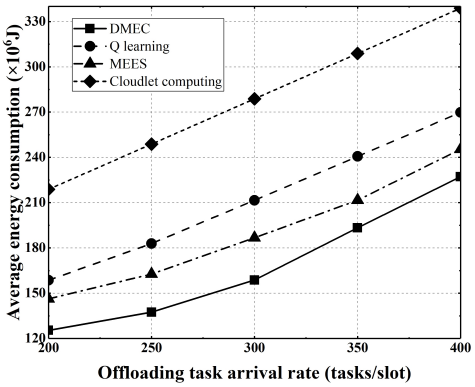



Fig. 6. Average energy consumption with different offloading task arrival rates.

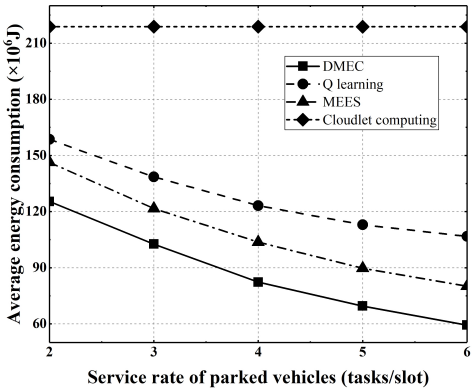



Fig. 7. Average energy consumption with different service rates of parked vehicles.

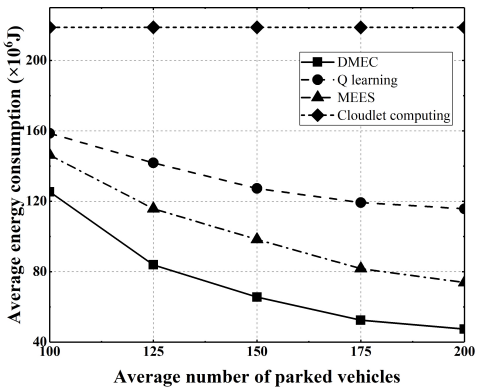



Fig. 8. Average energy consumption with different number of parked vehicles.

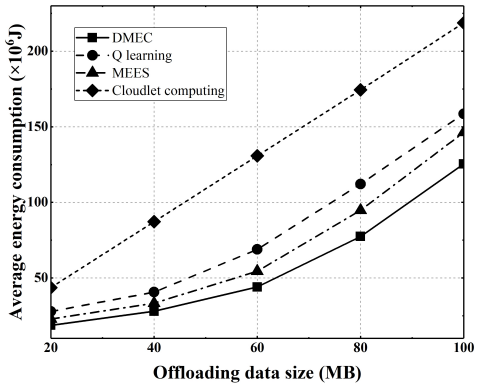



Fig. 9. Average energy consumption with different sizes of offloading tasks.

RSU. It can be observed that energy consumption raises as the task arrival rate increases. The trends of cloudlet computing and randomized strategy raise almost linearly, while the growth rate of the proposed DMEC algorithm increases slowly. When the task arrival rate increases from 100 to 400 per time slot, the energy consumption reduction ratio of the DMEC compared with cloudlet computing decreases from 43 percent to 33 percent, respectively. In addition, the reduction ratio decreases quickly with the increasing of the task arrival rate. The reasons are as follows: When the task arrival rate is relatively low (e.g., 100 or 200 per time slot), fog nodes (i.e., both parked and moving vehicles) can fulfill task loading as arrival rate increases. Since fog nodes consume less energy than cloudlet servers, average energy consumption grows slowly. However, when the number of increased tasks exceeds the burden of fog nodes, overloaded tasks need to be transferred to the cloudlet server for processing, increasing the growth rate of DMEC.

The results of average energy consumption based on different service rates of parked vehicles are shown in Fig. 7. When the service rate of parked vehicles increases, the processing capability of parked vehicles becomes powerful. As a result, there is a steady decline on average energy consumption with the increasing of the service rate. Since cloudlet is irrelevant to parked vehicles, the performance of cloudlet computing remains constant. The performance of DMEC is about 33 percent better than that of Q-learning. This is because Q-learning always chooses system actions in a greedy manner. On the contrary, DMEC applies trial-and-error search to balance exploration and exploitation. In addition, DMEC can learn from tagged historical data, increasing the accuracy of DQN.

Fig. 8 elaborates the performance of average energy consumption varying with average number of parked vehicles. The number of parked vehicles at each RSU is randomly generated by the uniform distribution centered on the average number, and average energy consumption is calculated as the performance indicator. Through comparison, we can observe that average energy consumption drops gradually with the increasing number of parked vehicles. This is because more parked vehicles share the burden of cloudlet servers with a relatively small amount of energy consumption, reducing the total energy consumption of the system. Especially, when computing resources of fog nodes are insufficient (e.g., the average number of parked vehicles is from 100 to 125), the average energy consumption decreases sharply. However, when the number of parked vehicles is relatively sufficient, this effect is not such obvious.

The performance of average energy consumption with different sizes of offloading tasks is illustrated in Fig. 9. The increasing of offloading data size mainly influences the energy consumption from transferring among RSUs and processing servers. Since fog nodes locate in the proximity of RSUs, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2019.2930521, IEEE Transactions on Cognitive Communications and Networking



performance of cloudlet computing raises faster than that of the proposed DMEC, which consumes less time to transfer offloading data to fog nodes. The larger the amount of data is, the more obvious the energy consumption can be reduced. However, when the offloading task is beyond the computation capability of vehicles (e.g., 80MB), the consumed energy of the DMEC algorithm increases quickly.

## VIII. Conclusion

This paper investigates energy efficient task offloading with DRL. We focus on a three-layer network model, where both parked vehicles and moving vehicles are leveraged as fog nodes based on queuing theory. By jointly considering load balance and delay constraint, we formulate the optimization problem to minimize energy consumption during traffic offloading. Due to its computation complexity, this problem is further divided into two stages: flow redirection and offloading decision. Edmonds-Karp and DRL based DMEC algorithms are respectively developed to solve the formulated problem. Finally, numerical results based on real-world traces of taxies in Shanghai (China) demonstrate that the Edmonds-Karp based algorithm can approach the performance gained by the exhaustive method, and decreases the energy consumption by 30 percent. For the second stage, the DMEC algorithm performs 35 and 60 percent better than Q-learning and cloudlet computing schemes, respectively.

## IX. Acknowledgments

This work is partially supported by National Nature Science Foundation of China under Grants 61671092 and 61771120, by China Postdoctoral Science Foundation under grant 2018T110210, by Fundamental Research Funds for the Central Universities under Grant DUT19JC18 and Grant DUT18JC09, by National Funding from the FCT - *Fundação para a Ciência e a Tecnologia*, through the UID/EEA/50008/2019 Project; by RNP, with resources from MCTIC, Grant No. 01250.075413/2018-04, under the *Centro de Referência em Radiocomunicações* - CRR project of the *Instituto Nacional de Telecomunicações* (Inatel), Brazil; and by Brazilian National Council for Research and Development (CNPq) via Grant No. 309335/2017-5.

## References

[1] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," *Traffic Safety Facts - Crash Stats*, 2015.

[2] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2432–2455, 2017.

[3] Z. Ning, X. Hu, Z. Chen, M. Zhou, B. Hu, J. Cheng, and M. S. Obaidat, "A cooperative quality-aware service access system for social internet of vehicles," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2506–2517, 2017.

[4] W. Hou, Z. Ning, and L. Guo, "Green survivable collaborative edge computing in smart cities," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1594–1605, 2018.

[5] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 87–93, 2019.

[6] Z. Ning, F. Xia, N. Ullah, X. Kong, and X. Hu, "Vehicular social networks: Enabling smart mobility," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 49–55, 2017.

[7] "Cisco global cloud index: Forecast and methodology, 2016–2021," *White Paper*, 2018.

[8] Z. Ning, F. Xia, X. Hu, Z. Chen, and M. S. Obaidat, "Social-oriented adaptive transmission in opportunistic Internet of smartphones," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 810–820, 2016.

[9] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, 2019.

[10] I. Delhi, "Automotive revolution and perspective towards 2030," *Auto Tech Review*, vol. 5, no. 4, pp. 20–25, 2016.

[11] S. Emprecha, W. Pattaraprakorn, V. Chutiprapat, and P. Bhasaputra, "The study on the effect of electric bus (non-fixed route) to energy consumption in thailand," in *IEEE ECTI-CON*, 2016, pp. 1–5.

[12] M. Chen, Y. Tian, G. Fortino, J. Zhang, and I. Humar, "Cognitive internet of vehicles," *Computer Communications*, vol. 120, pp. 58–70, 2018.

[13] D. B. Rawat, R. Alsabet, C. Bajracharya, and M. Song, "On the performance of cognitive Internet-of-vehicles with unlicensed user-mobility and licensed user-activity," *Computer Networks*, vol. 137, pp. 98–106, 2018.

[14] N. Cheng, N. Lu, N. Zhang, X. Zhang, X. S. Shen, and J. W. Mark, "Opportunistic WiFi offloading in vehicular environment: A game-theory approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1944–1955, 2016.

[15] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4568–4578, 2018.

[16] M. Tao, K. Ota, and M. Dong, "Foud: Integrating fog and cloud for 5G-enabled V2G networks," *IEEE Network*, vol. 31, no. 2, pp. 8–13, 2017.

[17] B. Tang, Z. Chen, G. Hefferman, S. Pei, W. Tao, H. He, and Q. Yang, "Incorporating intelligence in fog computing for big data analysis in smart cities," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2140–2150, 2017.

[18] L. T. Tan, R. Q. Hu, and L. Hanzo, "Twin-timescale artificial intelligence aided mobility-aware edge caching and computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, 2019.

[19] B. Mao, Z. M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Transactions on Computers*, vol. 66, no. 11, pp. 1946–1960, 2017.

[20] Z. Li, C. Wang, and C. J. Jiang, "User association for load balancing in vehicular networks: An online reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 8, pp. 2217–2228, 2017.

[21] Z. Ning, P. Dong, X. Wang, J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Transactions on Intelligent Systems and Technology*, vol. 25, p. 1, 2019.

[22] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10 190–10 203, 2018.

[23] Q. Qi and Z. Ma, "Vehicular edge computing via deep reinforcement learning," *arXiv preprint arXiv:1901.04290*, 2018.

[24] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, 2017.

[25] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[26] J. He, L. Cai, P. Cheng, and J. Pan, "Delay minimization for data dissemination in large-scale VANETs with buses and taxis," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 1939–1950, 2016.

[27] L. Klennrock, "Queueing systems volume 1: Theory," *New York*, 1975.

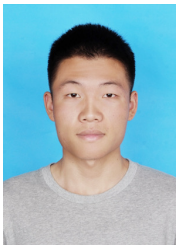
[28] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Reinforcement learning for resource provisioning in the vehicular cloud," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 128–135, 2016.

[29] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning." in *AAAI*, vol. 2. Phoenix, AZ, 2016, p. 5.

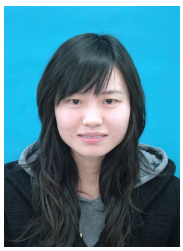
[30] Y. Wei, F. R. Yu, and M. Song, "Distributed optimal relay selection in wireless cooperative networks with finite-state markov channels," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 5, pp. 2149–2158, 2010.

[31] Z. Ning, J. Huang, X. Wang, J. Rodrigues, and L. Guo, "Mobile edge computing-enabled internet of vehicles: Toward energy-efficient scheduling," *IEEE Network*, 2019.

**Zhaolong Ning (M'14-SM'18)** received the M.S. and PhD degrees from Northeastern University, Shenyang, China. He was a Research Fellow at Kyushu University, Japan. He is an associate professor in the School of Software, Dalian University of Technology, China, and a Hong Kong scholar with The University of Hong Kong. He has published over 100 scientific papers in international journals and conferences. His research interests include Internet of vehicles, edge computing, and artificial intelligence.



**Peiran Dong** received B.S. degree from Dalian University of Technology, Dalian, China, in 2018. He is currently working toward the M.S. degree in the School of Software, Dalian University of Technology. His research interests include mobile edge computing, artificial intelligence and resource management.



**Xiaojie Wang** received the M.S. degree from Northeastern University, China, in 2011, and received the Ph.D. degree from Dalian University of Technology, China, in 2019. From 2011 to 2015, she was a software engineer in NeuSoft Corporation, China. She is a research associate in Lanzhou university. Her research interests are vehicular networks, edge computing and resource management. She has published over 30 scientific papers in the above areas.

**Lei Guo** received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2006. He is currently a Full Professor with Chongqing University of Posts and Telecommunications, Chongqing, China. He has authored or coauthored more than 200 technical papers in international journals and conferences. He is an editor for several international journals. His current research interests include communication networks, optical communications, and wireless communications.

**Joel J. P. C. Rodrigues (S'01-M'06-SM'06)** is currently a Professor and a Senior Researcher with the National Institute of Telecommunications, Brazil; a Senior Researcher with the Instituto de Telecomunicações, Portugal; and visiting professor at the Federal University of Piauí, Brazil. He is also the Leader of the Internet of Things Research Group (CNPq), Director for Conference Development - IEEE ComSoc Board of Governors, IEEE Distinguished Lecturer, a past Chair of the IEEE ComSoc Technical Committee on eHealth, a past Chair of the IEEE ComSoc Technical Committee on Communications Software. He has authored or co-authored over 750 papers in refereed international journals and conferences, three books, and two patents. He is the editor-in-chief of the International Journal on E-Health and Medical Communications and editorial board member of several high-reputed journals. He had been awarded several Outstanding Leadership and Outstanding Service Awards by the IEEE Communications Society and several best papers awards.

**Xiangjie Kong (M'13-SM'17)** received the BSc and PhD degrees from Zhejiang University, Hangzhou, China. He is currently an Associate Professor in School of Software, Dalian University of Technology, China. He has served as (Guest) Editor of several international journals, Workshop Chair or PC Member of a number of conferences. Dr. Kong has published over 100 scientific articles in international journals and conferences (with 70+ indexed by ISI SCIE). His research interests include intelligent transportation systems, mobile computing, and cyber-physical systems.

**Jun Huang (M'12-SM'16)** received the Ph.D. degree (with honor) from the Institute of Network Technology, Beijing University of Posts and Telecommunications, China, in 2012. He is a full professor of computer science with the Chongqing University of Posts and Telecommunications. He received the outstanding service award from ACM RACS 2017 and 2018, the runner-up of best paper award from ACM SAC 2014, and the best paper award from AsiaFI 2011. He has authored 100+ publications including papers in prestigious journal/conferences. His current research interests include network optimization and control, machine-to-machine communications, and the Internet of Things.

**Ricky Y. K. Kwok (F'14)** is a professor and Associate Vice-President at The University of Hong Kong, Hong Kong. He has been serving as an Associate Editor for the IEEE Transactions on Parallel and Distributed Systems. He also serves as a member of the Editorial Board for the International Journal of Sensor Networks, Journal of Parallel and Distributed Computing, and Peer-to-Peer (P2P) Computing. He is a Fellow of HKIE, IEEE, and IET. His recent research endeavors are mainly related to incentive, dependability, and security issues in wireless systems and P2P applications. He is also spending much time on task scheduling and mapping in contemporary parallel processing platforms, such as chip multiprocessors, dynamically reconfigurable systems, and clouds. He received his B.Sc. degree in computer engineering from HKU in 1991, the M.Phil. and Ph.D. degrees in computer science from the Hong Kong University of Science and Technology (HKUST) in 1994 and 1997, respectively.